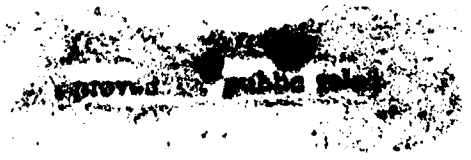
Learning DNF over the Uniform Distribution using a Quantum Example Oracle

Nader H. Bshouty* Jeffrey Jackson

September 1994
CMU-CS-94-190

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

*University of Calgary.



The first author's research was supported by NSERC. The second author's research was sponsored by the National Science Foundation under Grant No. CCR-9119319.

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government.

Keywords: quantum computation, computational learning theory, DNF (disjunctive normal form), quantum example oracle, classification noise, statistical query learning

Abstract

We generalize the notion of PAC learning from an example oracle to a notion of efficient learning on a quantum computer using a quantum example oracle. This quantum example oracle is a natural extension of the traditional PAC example oracle, and it immediately follows that all PAC-learnable function classes are learnable in the quantum model. Furthermore, we obtain positive quantum learning results for classes that are not known to be PAC learnable. Specifically, we show that DNF is efficiently learnable with respect to the uniform distribution by a quantum algorithm using a quantum example oracle. While it was already known that DNF is uniform-learnable using a membership oracle, the quantum example oracle is provably less powerful than a membership oracle. We also generalize the notion of classification noise to the quantum setting and show that the quantum DNF algorithm learns even in the presence of such noise. This result contrasts with a recent negative result for DNF in the statistical query model of learning from noisy data.

1 Introduction

Recently, there has been increasing interest in the question of whether or not quantum physical effects can be used to solve problems that appear to be computationally difficult using traditional methods. In this paper, we apply quantum methods to questions in computational learning theory. In particular, we focus on the problem of learning—from examples alone—the class DNF of polynomial-size Disjunctive Normal Form expressions.

The DNF learning problem has a long history. Valiant [18] introduced the problem and gave efficient algorithms for learning certain subclasses of DNF. Since then, learning algorithms have been developed for a number of other subclasses of DNF [13, 4, 3, 11, 2, 1, 7, 16, 9] and recently for the unrestricted class of DNF expressions [6, 12], but almost all of these results—and in particular the results for the unrestricted class—use membership queries (the learner is told the output value of the target function on learner-specified inputs). This has left open the question of to what extent membership queries are necessary for DNF learning, even in models where the learner is only required to produce a hypothesis that weakly approximates the target DNF expression with respect to the uniform distribution (definitions are given in the next section).

We show that DNF is efficiently learnable with respect to the uniform distribution by a quantum algorithm that receives its information about the target function from a *quantum example oracle*. This oracle generalizes the traditional PAC example oracle in a natural way. Specifically, the quantum oracle $QEX(f, D)$ is a traditional PAC example oracle $EX(f, D)$ except that $QEX(f, D)$ produces the example $\langle x, f(x)\rangle$ with amplitude $\sqrt{D(x)}$ rather than with probability $D(x)$. We show that, with respect to the uniform distribution, a quantum example oracle can be simulated by a membership oracle but not vice versa. Thus our result can be viewed as evidence that DNF is learnable without the full power of membership queries.

Furthermore, we generalize the notion of classification noise and show that our algorithm learns DNF even if the quantum example oracle exhibits such noise. This is particularly interesting in light of recent results of Blum et al. [6] showing that DNF is not learnable in the statistical query (SQ) model. Because SQ learning is conjectured to be equivalent to the model of PAC learning with classification noise, our result is evidence that quantum learning algorithms may be better able to handle noise than traditional algorithms.

To obtain our quantum DNF learning algorithm, we modify the recent Harmonic Sieve algorithm (HS) for learning DNF with respect to uniform using membership queries [12]. In fact, HS properly learns the larger class $\widehat{PT}_1$ of functions expressible as a threshold of a polynomial number of parity functions, and our algorithm properly learns this class as well. The Harmonic Sieve uses membership queries to locate parity functions that correlate well with the target function with respect to various near-uniform distributions. The heart of our result is showing that these parities can be located efficiently by a quantum algorithm using only a quantum example oracle.

2 Definitions and Notation

2.1 Functions and Function Classes

We will be interested in the learnability of sets (*classes*) of Boolean functions. The Boolean functions we consider are, unless otherwise noted, of the type $f : \{0,1\}^n \to \{0,1\}$ for fixed positive values of n. We call $\{0,1\}^n$ the *instance space* of f, an element x in the instance space an *instance*, and the pair $\langle x, f(x)\rangle$ an *example* of f. We denote by x_i the ith bit of instance x.

Intuitively, a learning algorithm should be allowed to run in time polynomial in the complexity of the function f to be learned; we will use the *size* of a function as a measure of its complexity. The size measure will depend on the function class to be learned. In particular, each function class $\mathcal{F}$ that we study implicitly defines a natural class $\mathcal{R}_{\mathcal{F}}$ of representations of the functions in $\mathcal{F}$. We define the *size of a function* $f \in \mathcal{F}$ as the minimum, over all $r \in \mathcal{R}_{\mathcal{F}}$ such that r represents f, of the size of r, and we define below the size measure for each representation class of interest.

A DNF expression is a disjunction of terms, where each term is a conjunction of literals and a literal is either a variable or its negation. The *size of a DNF expression* r is the number of terms in r. The DNF function class is the set of all functions that can be represented as a DNF expression of size polynomial in n.

Following Bruck [8], we use $\widehat{PT}_1$ to denote the class of functions on $\{0,1\}^n$ expressible as a depth-2 circuit with a majority gate at the root and polynomially-many parity gates at the leaves. All gates have unbounded fanin and fanout one. The *size of a* $\widehat{PT}_1$ *circuit* r is the number of parity gates in r.

2.2 Quantum Turing Machines

We now review the model of quantum computation defined by Bernstein and Vazarani [5]. First we define how the specification (program) of a *quantum Turing machine* (QTM) is written down. Then we describe how a QTM operates.

The specification of a QTM is exactly the same as the specification of a probabilistic TM, except the transition probabilities between PTM configurations are replaced in a QTM specification with complex-valued numbers (*amplitudes*) that satisfy a certain *well-formedness* property. We define well-formedness as follows. For a QTM M, let R_M be the (infinite-dimensional) matrix where each row and each column is labeled with a machine configuration (c_r and c_c, respectively) and each entry in R_M is the amplitude assigned by M to the transition from configuration c_c to c_r. Then M satisfies the well-formedness property if R_M is unitary ($R_M^\dagger R_M = R_M R_M^\dagger = I$, where $R_M^\dagger$ is the conjugate transpose of R_M). A QTM specification also contains a set of states (including all of the final states) in which an *Obs* operation is performed; we define this operation below.

To describe the operation of a QTM, we use the notion of a *superposition* of configurations. For example, consider a probabilistic Turing machine M' that at step i flips a fair coin and chooses to transition to one of two configurations c_1 and c_2. While we would generally think of M' as being in exactly one of these configurations at step $i+1$, we can equivalently think of M' as being in *both* states, each with probability 1/2. Continuing in this fashion, for

each step until M' terminates we can think of M' as being in a superposition of states, each state with an associated probability. After M' takes its final step, each of its final states will have some associated probability (we assume without loss of generality that all computation paths in M' have the same length). If M' now "chooses" to be in one of these final states σ_f randomly according to the induced probability distribution on final states, then the probability of being in σ_f is exactly the same in this model as it is in the traditional PTM model.

In summary, we can view a PTM M' as being in a superposition of configurations at each step, where a superposition is represented by a vector of probabilities, one for each possible configuration of M'. Likewise, we view a QTM M as being in a superposition of configurations at each step, but now the superposition vector contains an amplitude for each possible configuration of M. The initial superposition vector in both cases is the all-zero vector except for a single 1 in the position corresponding to the initial configuration of the machine. Note that each step of a PTM M' can be accomplished by multiplying the current superposition vector by a matrix $R_{M'}$ which is defined analogously with R_M above. In the same way, each step of a QTM M is accomplished by multiplying its current superposition vector by R_M. The difference between the machines comes at the point(s) where M "chooses" to be in a single configuration rather than in a superposition of configurations. M does this (conceptually) by transitioning to a superposition of configurations all of which are in one of the *Obs* states mentioned above. The superposition vector is then changed so that a single configuration has amplitude 1 and all others are 0. This is exactly analogous to the PTM M' choosing its final state, except that the probability of choosing each configuration c_i is now the *square* of the magnitude of the amplitude associated with c_i in M's current superposition vector. (We formalize the definition of *Obs* below.)

We adopt Simon's notation [17] and write

$$\sum_x a_x|x\rangle$$

to denote a superposition of configurations x each having amplitude a_x. While in general this sum is over all possible configurations of the QTM, when we use this notation it will be the case that all of the configurations having nonzero amplitude are in the same state and have the tape head at the same position. In this case, the configurations x are only distinguished by their tape contents, so we will treat x as if it is merely the tape content and ignore other configuration parameters.

Given this notation, we formally define the *Obs* operation as follows. Let $b \in \{0,1\}$. Then

$$Obs\left(\sum_{bx} a_{bx}|bx\rangle\right) = \begin{cases} \sum_x \frac{a_{0x}}{\sum_y |a_{0y}|^2}|0x\rangle & \text{with probability } \sum_x |a_{0x}|^2 \\ \sum_x \frac{a_{1x}}{\sum_y |a_{1y}|^2}|1x\rangle & \text{with probability } \sum_x |a_{1x}|^2. \end{cases}$$

Note that by permuting bits of the tape and performing successive *Obs* operations we can simulate the informal definition of *Obs* given earlier. We say that a language L is in BQP if there exists a QTM M such that, at the end of a polynomial number of steps by M, an *Obs* fixes the first tape cell to 1 with probability at least 2/3 if the input is in L and fixes it to 0 with probability at least 2/3 otherwise. We will also sometimes think of an *Obs* as simply computing the probability that the first cell will be fixed to 1.

2.3 Learning Models

We being by defining the well-known PAC learning model and then generalize this to a quantum model of learning. First, we define several supporting concepts. Given a function f and probability distribution D on the instance space of f, we say that function h is an *ϵ-approximator for f with respect to D* if $\Pr_D[h = f] \geq 1 - \epsilon$. An *example oracle for f with respect to D* ($EX(f, D)$) is an oracle that on request draws an instance x at random according to probability distribution D and returns the example $\langle x, f(x)\rangle$. A *membership oracle for f* ($MEM(f)$) is an oracle that given any instance x returns the value $f(x)$. Let $\mathcal{D}_n$ denote a nonempty set of probability distributions on $\{0,1\}^n$. Any set $\mathcal{D} = \cup_n \mathcal{D}_n$ is called a *distribution class*. We let $\mathcal{U}_n$ represent the uniform distribution on $\{0,1\}^n$ and call $\mathcal{U} = \cup_n \mathcal{U}_n$ simply the *uniform distribution*.

Now we formally define the Probably Approximately Correct (PAC) model of learnability [18]. Let ϵ and δ be positive values (called the *accuracy* and *confidence* of the learning procedure, respectively). Then we say that the function class $\mathcal{F}$ is *(strongly) PAC learnable* if there is an algorithm $\mathcal{A}$ such that for any ϵ and δ, any $f \in \mathcal{F}$ (the *target function*), and any distribution D on the instance space of f (the *target distribution*), with probability at least $1 - \delta$ algorithm $\mathcal{A}(EX(f, D), \epsilon, \delta)$ produces an ϵ-approximation for f with respect to D in time polynomial in n, the size of f, $1/\epsilon$, and $1/\delta$. The probability that $\mathcal{A}$ succeeds is taken over the random choices made by EX and $\mathcal{A}$ (if any). We generally drop the "PAC" from "PAC learnable" when the model of learning is clear from context.

Next, we consider learning using a QTM. First, note that each call to the traditional PAC example oracle $EX(f, D)$ can be viewed as defining a superposition of 2^n configurations, each containing a distinct $\langle x, f(x)\rangle$ pair and having probability of occurrence $D(x)$. We generalize this to the quantum setting in a natural way. A *quantum example oracle for f with respect to D* ($QEX(f, D)$) is an oracle running coherently with a QTM M that changes M's tape $|y\rangle$ to

$$\sum_x \sqrt{D(x)}|y, x, f(x)\rangle.$$

That is, QEX defines a superposition of 2^n configurations much as EX does, but QEX assigns each configuration an amplitude $\sqrt{D(x)}$. Note that a call to QEX followed by an Obs operation is equivalent to a call to EX. We say that $\mathcal{F}$ is *quantum learnable* if $\mathcal{F}$ is PAC learnable by a QTM M using a quantum example oracle. Because every efficient TM computation can be simulated efficiently by a QTM [5] and because EX can be simulated by QEX, we have that every PAC-learnable function class is also quantum learnable.

We will consider several variations on the basic PAC and quantum models. Let $\mathcal{M}$ be any model of learning (e.g., PAC). If $\mathcal{F}$ is $\mathcal{M}$-learnable by an algorithm $\mathcal{A}$ that requires a membership oracle then $\mathcal{F}$ is *$\mathcal{M}$-learnable using membership queries*. If $\mathcal{F}$ is $\mathcal{M}$-learnable for $\epsilon = 1/2 - 1/p(n, s)$, where p is a fixed polynomial and s is the size of f, then $\mathcal{F}$ is *weakly $\mathcal{M}$-learnable*. We say that $\mathcal{F}$ *is $\mathcal{M}$-learnable by $\mathcal{H}$* if $\mathcal{F}$ is $\mathcal{M}$-learnable by an algorithm $\mathcal{A}$ that always outputs a function $h \in \mathcal{H}$. If $\mathcal{F}$ is $\mathcal{M}$-learnable by $\mathcal{F}$ then we say that $\mathcal{F}$ is *properly $\mathcal{M}$*-learnable. Finally, note that the PAC model places no restriction on the example distribution D; we sometimes refer to such learning models as *distribution-independent*. If $\mathcal{F}$ is $\mathcal{M}$-learnable for all distributions D in distribution class $\mathcal{D}$ then $\mathcal{F}$ is *$\mathcal{M}$-learnable with respect to $\mathcal{D}$*.

2.4 The Fourier Transform

For each bit vector $a \in \{0,1\}^n$ we define the function $\chi_a : \{0,1\}^n \to \{-1,+1\}$ as

$$\chi_a(x) = (-1)^{\sum_{i=1}^n a_i x_i} = 1 - 2\left(\sum_{i=1}^n a_i x_i \bmod 2\right).$$

That is, $\chi_a(x)$ is the boolean function that is 1 when the parity of the bits in x indexed by a is even and is -1 otherwise. With inner product defined by[1] $\langle f, g\rangle = \mathbf{E}[fg]$ and norm defined by $\|f\| = \sqrt{\mathbf{E}[f^2]}$, $\{\chi_a \mid a \in \{0,1\}^n\}$ is an orthonormal basis for the vector space of real-valued functions on the Boolean cube $\mathbf{Z}_2^n$. That is, every function $f : \{0,1\}^n \to \mathbf{R}$ can be uniquely expressed as a linear combination of parity functions:

$$f = \sum_a \hat{f}(a)\chi_a,$$

where $\hat{f}(a) = \mathbf{E}[f\chi_a]$. We call the vector of coefficients $\hat{f}$ the *Fourier transform* of f. Note that for Boolean f, $\hat{f}(a)$ represents the correlation of f and χ_a with respect to the uniform distribution. Also note that $\hat{f}(\emptyset) = \mathbf{E}[f\chi_\emptyset] = \mathbf{E}[f]$, since $\chi_\emptyset$ is the constant function $+1$.

Parseval's identity states that for every function f, $\mathbf{E}[f^2] = \sum_a \hat{f}^2(a)$. For Boolean f it follows that $\sum_a \hat{f}^2(a) = 1$. More generally, it can be shown that for any functions f and g, $\mathbf{E}[fg] = \sum_a \hat{f}(a)\hat{g}(a)$.

3 DNF Learning

In this section we present our primary result, that DNF is quantum learnable with respect to the uniform distribution. Our result builds on the HS algorithm for learning DNF with respect to the uniform distribution using membership queries [12]. The HS algorithm depends on a key fact about DNF expressions: for every DNF f with s terms and for every distribution D there exists a parity χ_a such that $|\mathbf{E}_D[f\chi_a]| \geq 1/(2s+1)$ [12]. It follows immediately from this fact that for every DNF f and distribution D there is a parity χ_a that is a weak approximator to f with respect to D. Furthermore, for many probability distributions (e.g., distributions such that $D(x) \leq p(1/\epsilon)/2^n$ for all x and for p a fixed polynomial), an algorithm of Kushilevitz and Mansour [15] can be used to efficiently find such a χ_a. The Kushilevitz-Mansour algorithm is the only aspect of HS that requires membership queries. Finally, an algorithm of Freund [10] is employed to boost the Kushilevitz-Mansour weak learning algorithm into a strong learner.

The HS algorithm and its primary subroutine WDNF are sketched in Figures 1 and 2 (c_1 and c_2 represent fixed constants). The HS algorithm runs for at most k steps, or stages. $r_i(x)$ represents the number of weak hypotheses w_j among those hypotheses produced before stage i that are "right" on x. At each stage i, the boosting algorithm implicitly defines a distribution $D_i(x)$ based on $r_i(x)$; this is the distribution for which the weak learner is expected to produce a weak hypothesis. We can generate x's according to this distribution

[1] Expectations and probabilities here and elsewhere are with respect to the uniform distribution over the instance space unless otherwise indicated.

Invocation: $h \leftarrow \texttt{HS}(n, s, MEM(f), \epsilon, \delta)$
Input: n; s = size of DNF f; $MEM(f)$; $\epsilon > 0$; $\delta > 0$
Output: with probability at least $1 - \delta$ (over random choices made by HS), HS returns h such that $\Pr[f = h] \geq 1 - \epsilon$

1. $k \leftarrow c_1 s^2 \log(1/\epsilon)$
2. $B(j; n, p) \equiv \binom{n}{j} p^j (1-p)^{n-j}$
3. $\beta_r^i \equiv B(\lfloor k/2 \rfloor - r; k - i - 1, 1/2 + 1/(4s+2))$ if $i - k/2 < r \leq k/2$, $\beta_r^i \equiv 0$ otherwise
4. $\alpha_r^i \equiv \beta_r^i / \max_{r=0,\ldots,i-1}\{\beta_r^i\}$.
5. $w_0 \leftarrow \texttt{WDNF}(n, s, MEM(f), \mathcal{U}_n, \delta/2k)$
6. **for** $i \leftarrow 1, \ldots, k-1$ **do**
7. $\quad r_i(x) \equiv |\{0 \leq j < i \mid w_j(x) = f(x)\}|$
8. $\quad accept \leftarrow \textbf{Est}(\mathbf{E}_x[\alpha^i_{r_i(x)}], c_2\epsilon^2/3, \delta/2k)$
9. $\quad$ **if** $accept \leq 2c_2\epsilon^2/3$ **then**
10. $\quad\quad k \leftarrow i$
11. $\quad\quad$ **break do**
12. $\quad$ **endif**
13. $\quad \tilde{D}_i(x) \equiv \alpha^i_{r_i(x)} / 2^n accept$
14. $\quad w_i \leftarrow \texttt{WDNF}(n, s, MEM(f), \tilde{D}_i(x), \delta/2k)$
15. **enddo**
16. **return** $h(x) \equiv MAJ(w_0(x), w_1(x), \ldots, w_{k-1}(x))$

Figure 1: Harmonic Sieve algorithm for learning DNF.

as follows. Choose an instance x uniformly at random and flip a coin that comes up heads with probability $\alpha^i_{r_i(x)}$ (α is a scaled binomial distribution). If the coin comes up heads, output x. Otherwise, select a new x uniformly at random and flip the coin again. Repeat this process until some x is output.

Thus

$$D_i(x) = \frac{\alpha^i_{r_i(x)}}{\sum_y \alpha^i_{r_i(y)}}. \tag{1}$$

In order for the Kushilevitz-Mansour algorithm to find a good weak approximator with respect to distribution D_i, we need to be able to closely approximate $D_i(x)$ for every value of x. The function $\tilde{D}_i$ is HS's estimate of D_i. Note that because of the bound on the variable *accept* and the accuracy with which *accept* estimates $\mathbf{E}_x[\alpha^i_{r_i(x)}]$, with high probability $\mathbf{E}_x[\alpha^i_{r_i(x)}] = c_3 accept$ for fixed $c_3 \in [1/2, 3/2]$, and thus $\tilde{D}_i(x) = c_3 D_i(x)$ for all x.

We have omitted the details of line 2 of **WDNF** because this is the main point at which our quantum algorithm will differ from HS. Rather than using membership queries to locate the required parity χ_a, the new algorithm will use a quantum example oracle. Both algorithms depend on a key fact about DNF expressions: for every DNF f with s terms and for every distribution D there exists a parity χ_a such that $|\mathbf{E}_D[f\chi_a]| \geq 1/(2s+1)$ [12]. We will show how to find such a parity χ_a for any distribution D_i simulated by HS given access only to a quantum example oracle.

Invocation: $w_i \leftarrow$ **WDNF**$(n, s, MEM(f), cD, \delta)$
Input: n; s = size of DNF f; $MEM(f)$; cD, an oracle that given x returns $c \cdot D(x)$, where c is a constant in $[1/2, 3/2]$ and D is a probability distribution on $\{0,1\}^n$; $\delta > 0$
Output: with probability at least $1 - \delta$ (over random choices made by **WDNF**), **WDNF** returns h such that $\Pr_D[f = h] \geq 1/2 + 1/(4s + 2)$

1. $g(x) \equiv c2^n f(x)D(x)$
2. find (using membership queries and with probability at least $1 - \delta$) χ_a such that $|\mathbf{E}_D[g\chi_a]| \geq 1/2s + 1$
3. **return** $h(x) \equiv \text{sign}(\mathbf{E}_D[g\chi_a]) \cdot \chi_a(x)$

Figure 2: **WDNF** subroutine called by **HS**.

Specifically, consider the call to **WDNF** at line 14 of **HS** for a fixed i, and for notational convenience let $D \equiv D_i$ and $\alpha(x) \equiv \alpha^i_{r_i(x)}$. Then with high probability there is a $c_3 \in [1/2, 3/2]$ such that for all χ_a,

$$\sum_x f(x)\chi_a(x)\tilde{D}_i(x) = c_3\mathbf{E}_D[f\chi_a] = \mathbf{E}[\alpha f\chi_a]/accept,$$

and thus for some χ_a,

$$|\mathbf{E}[\alpha f\chi_a]| \geq \frac{c_2\epsilon^2}{3(2s+1)}.$$

Our goal will be to find such a χ_a using only a quantum example oracle for f. From this χ_a we can produce a weak approximator for f, as illustrated by **WDNF**. Conceptually, to find such a χ_a we will run a quantum program that will sample the χ_a's with probability proportional to $\mathbf{E}^2[\alpha f\chi_a]$. The technique we use to perform this sampling is similar to an algorithm of Bernstein and Vazarani that samples the χ_a's with probability $\hat{f}^2(a) = \mathbf{E}^2[f\chi_a]$. However, there are two difficulties with using their technique directly. First, their algorithm uses calls to the function f (membership queries), and we want an algorithm that uses only quantum example queries. Second, their technique works for Boolean functions, but $\alpha \cdot f$ is not Boolean. The following lemma addresses the first difficulty.

Lemma 1 *There is a quantum program* **QSAMP** *that, given any quantum example oracle* $QEX(f)$*, returns* χ_a *with probability* $\hat{f}^2(a)/2$.

Proof: **QSAMP** begins by calling $QEX(f)$ on a blank tape to get the superposition

$$\frac{1}{2^{n/2}}\sum_x |x, f(x)\rangle.$$

QSAMP next replaces $f(x)$ with $(1 - f(x))/2$ (call this $f'(x)$); note that $(-1)^{f'(x)} = f(x)$. Then we will apply a Fourier operator F to the entire tape contents. We define F as

$$F(|a\rangle) \equiv \frac{1}{2^{n/2}}\sum_y (-1)^{a \cdot y}|y\rangle$$

where $|a| = n$. This operation can be performed in n steps by a quantum Turing machine [5]. Also recall that $(-1)^{a \cdot y} \equiv \chi_a(y) \equiv \chi_y(a)$. Thus applying F gives us

$$\begin{aligned} F\left(\frac{1}{2^{n/2}} \sum_x |x, f'(x)\rangle\right) &= \frac{1}{2^{n/2}} \sum_x F(|x, f'(x)\rangle) \\ &= \frac{1}{2^{n+1/2}} \sum_{x,y,z} (-1)^{x \cdot y} (-1)^{f'(x)z} |y, z\rangle \\ &= \frac{1}{\sqrt{2}} \sum_y \mathbf{E}_x[\chi_y(x) f(x)] |y, 1\rangle + \frac{1}{\sqrt{2}} \sum_y \mathbf{E}_x[\chi_y(x)] |y, 0\rangle \\ &= \frac{1}{\sqrt{2}} \sum_y \hat{f}(y) |y, 1\rangle + \frac{1}{\sqrt{2}} |\bar{0}, 0\rangle \end{aligned}$$

where $|y| = |x| = n$ and $|z| = 1$ and the final line follows by orthonormality of the parity basis. An *Obs* operation at this point produces $|y, 1\rangle$ with probability $\hat{f}^2(y)/2$, as desired. □

However, we want to sample the parities according to the coefficients of the non-Boolean function αf. We will do this indirectly by sampling over individual bits of the function. First, note that we can limit the accuracy of α and still compute an adequate approximation to $\mathbf{E}[\alpha f \chi_a]$. That is, since $0 \leq \alpha(x) \leq 1$ for all x, for some $d = O(\log(s/\epsilon))$ and $\theta(x) = \lfloor 2^d \alpha(x) \rfloor 2^{-d}$ we have

$$|\mathbf{E}[\theta f \chi_a]| \geq |\mathbf{E}[\alpha f \chi_a]| - \frac{c_2 \epsilon^2}{12(2s+1)}$$

for all χ_a. Furthermore, every $\theta \leq 1$ can be written as

$$\theta = \theta_1 2^{-1} + \theta_2 2^{-2} + \cdots + \theta_d 2^{-d} + k 2^{-d}$$

where $\theta_i \in \{-1, +1\}$ and $k \in \{-1, 0, 1\}$. Thus

$$|E[\theta f \chi_a]| \leq \max_j |E[\theta_j f \chi_a]| + \frac{1}{2^d}.$$

Therefore, if χ_a is a good approximator to f then there is some j such that $|E[\theta_j f \chi_a]|$ is larger than $1/p_1(s, 1/\epsilon)$ for a fixed polynomial p_1. Furthermore, the number of such χ_a's for each j is bounded by a $p_1^2(s, 1/\epsilon)$ by Parseval's.

Thus a quantum algorithm for learning DNF with respect to uniform is obtained by modifying HS as follows. First, procedure **Est** (line 8 of HS) will now use $QEX(f)$ to simulate the example oracle $EX(f)$ in order to estimate expected values. Second, in order to find a good weak approximator (line 2 of WDNF) we will use the quantum approach outlined above. That is, for each value of j we will sample the χ_a's using a probability equal to $E^2[\theta_j f \chi_a]/2$. We do this by running a modified QSAMP that, after calling $QEX(f)$, replaces $f(x)$ with $\theta_j(x) \cdot f(x)$. This is a reversible operation because x is still on the tape and $\theta_j^2(x) = 1$ for all x. If we perform this sampling $2d \log(1/\delta)$ times for each value of j then with probability at least $1 - \delta$ one of the χ_a's returned will be a good weak approximator. Finally, we will simulate $EX(f)$ in order to test whether or not a given χ_a is a good weak approximator. This gives us

Theorem 2 *DNF is quantum learnable with respect to uniform.*

Also, $\widehat{PT}_1$, the class of functions expressible as a threshold of a polynomial number of parity functions, has the property that every $\widehat{PT}_1$ can be weakly approximated by a parity with respect to any fixed distribution D [12]. This was the only property of DNF that we used in the above arguments. Therefore

Theorem 3 $\widehat{PT}_1$ *is quantum learnable with respect to uniform.*

4 Membership Oracle vs. Quantum Example Oracle

In practice, it is not clear how a quantum example oracle could be constructed without using a membership oracle. Furthermore, because a QTM uses interference over an entire superposition to perform its computations, it might seem that perhaps there is some way to simulate a membership oracle given only a quantum example oracle by choosing a clever interference pattern. In this section we show that this is not the case.

Definition 4 *We say that* membership queries can be quantum-example simulated for function class $\mathcal{F}$ *if there exists a BQP algorithm* $\mathcal{A}$ *and a distribution* D *such that for all* $f \in \mathcal{F}$ *and all* x*, running* $\mathcal{A}$ *on input* x *with quantum example oracle* $QEX(f, D)$ *produces* $f(x)$*.*

Theorem 5 *Membership queries cannot be quantum-example simulated for DNF.*

Before proving this theorem, we develop some intuition. Consider two functions f_0 and f_1 that differ in exactly one input x. Then the superpositions returned by $QEX(f_0, D)$ and $QEX(f_1, D)$ are very similar for "almost all" D. In particular, if we think of superpositions as vectors in an inner produce space of dimension 2^n, then there is in general an exponentially small angle between the superpositions generated by these two oracles. This angle will not be changed by unitary transformations. So in general, an observation will be unable to detect a difference between the superpositions produced by $QEX(f_0, D)$ and $QEX(f_1, D)$. Therefore a BQP algorithm with only a quantum example oracle $QEX(f_i, D)$, $i \in \{0, 1\}$, will be unable to correctly answer a membership query on x for both f_0 and f_1.

We now present two lemmas that will help us to formalize this intuition.

Lemma 6 *Let* $\mathcal{A}$ *be a quantum algorithm that makes at most* t *calls to* $QEX(f, D)$*. Then there is an equivalent quantum program (modulo a polynomial slowdown) that makes all* t *calls at the beginning of the program.*

Proof: Let H be a unitary matrix representing an arbitrary move μ by the QTM M. Then if M is initially in a superposition $\sum_y a_y|y\rangle$, performs the move μ, and then calls $QEX(f, D)$, the resulting superposition will be

$$\sum_{x,y,z} \sqrt{D(x)} a_y H[z, y]|z, x\rangle,$$

where $|y| \leq |z| \leq |y| + 1$ (we are assuming without loss of generality a standard bit-string encoding of configurations). But notice that there is a machine M' that first calls QEX,

shifts x one cell to the right (if necessary), and then simulates the move μ. M' is at most polynomially slower than M and produces the same tape configuration given above. A simple inductive argument completes the proof. □

Before presenting the next lemma, we need several definitions.

Definition 7 *Define Obs over* any *linear combination of configurations (i.e., we no longer require that the sum of squared amplitudes be 1) as*

$$Obs\left(\sum_x u_x|x\rangle\right) = \sum_{x:x_1=1} |u_x|^2.$$

Define the length *of a linear combination of configurations* $S = \sum_x u_x|x\rangle$ *to be* $\|S\| = \sum_x |u_x|^2$. *For any linear combination of configurations* S *we define for* $i \in \{0,1\}$

$$S^{(i)} = \sum_{x:x_1=i} u_x|x\rangle.$$

Lemma 8 *Let* S_1 *and* S_2 *be superpositions and let* S *be any linear combination of configurations. Let* W *be any quantum operation. Then*

1. $Obs(S) \leq \|S\|$.
2. $\|WS\| = \|S\|$.
3. $|Obs(S_1) - Obs(S_2)| \leq Obs(S_1 - S_2)$.

Proof: For 1. we have

$$Obs(S) = \|S^{(1)}\| \leq \|S^{(0)}\| + \|S^{(1)}\| = \|S\|.$$

2. follows from the fact that W is a unitary operation that preserves length.

To prove 3. we have

$$\begin{aligned} |Obs(S_1) - Obs(S_2)| &= \left|\|S_1^{(1)}\| - \|S_2^{(1)}\|\right| \\ &\leq \|S_1^{(1)} - S_2^{(1)}\| \\ &= Obs(S_1 - S_2). \end{aligned}$$

□

Proof of Theorem 5: By Lemma 6, we can assume that all of the calls to QEX occur at the beginning of the program. Now suppose M with $QEX_{f,D}$ can quantum-simulate MEM_f. Take $f(x) = 0$ and $g(x) = x_1^{c_1} \wedge ... \wedge x_n^{c_n}$ where $x^d = 1$ if and only $x = d$. The second function is zero in all points except in $c = (c_1, \ldots, c_n)$. We want to use the simulator to find $h(c)$. The simulator will first make t calls to $QEX_{h,D}$ for $h \in \{f, g\}$ giving

$$S_h = \sum_{z_1,\ldots,z_t} \sqrt{D(z_1)\cdots D(z_t)}|z_1, h(z_1), \ldots, z_t, h(z_t)\rangle.$$

After that, the computation for f and g is the same. The superpositions for $h = f$ and $h = g$ differ only in configurations

$$|z_1, h(z_1), \ldots, z_t, h(z_t)\rangle$$

where one of the z_i is c.

Therefore

$$S_f = S_g + E_{f,g},$$

where

$$\begin{aligned} E_{f,g} = & \sum_{(\exists i) z_i = c} \sqrt{D(z_1) \cdots D(z_t)} |z_1, f(z_1), \ldots, z_t, f(z_t)\rangle - \\ & \sum_{(\exists i) z_i = c} \sqrt{D(z_1) \cdots D(z_t)} |z_1, g(z_1), \ldots, z_t, g(z_t)\rangle. \end{aligned}$$

If W is the computation after the oracle calls then we observe $Obs(WS_f)$ and $Obs(WS_g)$ for f and g, respectively. By Lemma 8

$$\begin{aligned} |Obs(WS_f) - Obs(WS_g)| &\leq Obs(WE_{f,g}) \\ &\leq \|WE_{f,g}\| \\ &= \|E_{f,g}\| \\ &= 2 \sum_{(\exists i) z_i = c} \left(\sqrt{D(z_1) \cdots D(z_t)}\right)^2 \\ &= 2(1 - (1 - D(c))^t) \\ &\leq 2tD(c). \end{aligned}$$

For almost all points c, $D(c) < 1/poly(n)$ for any $poly(n)$. For all such c our observation is indistinguishable. □

Note that while there are restrictions on the unitary matrix representing the transitions of a quantum Turing machine, we did not rely on these restrictions in the proof of this theorem. Thus we have actually proved the stronger result that, even given the ability to apply arbitrary unitary operations to superpositions, it is not possible to simulate membership queries in polynomial time given only a quantum example oracle. On the other hand, it is a simple matter to simulate a uniform quantum example oracle with membership queries.

Lemma 9 *For every Boolean function f, $QEX(f,\mathcal{U})$ can be simulated by $MEM(f)$.*

Proof: $QEX(f,\mathcal{U})$ can be simulated by applying the Fourier transform F to the tape $|\bar{0}\rangle$ and applying f. □

Thus a membership oracle for f is strictly more powerful than a uniform quantum example oracle for f.

5 Learning Noisy DNF's

Recently, Blum et al. [6] have shown that DNF is not learnable with respect to uniform in the statistical query (SQ) model of learning. The SQ model is apparently a good model of

learning with classification noise: if a function class is SQ learnable then it is learnable with classification noise, and every function class that is known to be learnable from such noise is also known to be SQ learnable [14]. Therefore, the fact that DNF is hard to learn in the SQ model would seem to be strong evidence that DNF is not PAC learnable from a noisy example oracle.

However, in this section we show that DNF is quantum learnable with respect to uniform using a noisy quantum example oracle. We define such an oracle as follows: given a noise rate η, a *noisy quantum example oracle for* f, $QEX^{\eta}(f, D)$, is exactly like the quantum example oracle $QEX(f, D)$ except that each of the example labels is reversed with probability η. Each call to the $QEX^{\eta}(f, D)$ chooses which labels to flip independently of all previous calls. We say that a function class is *learnable using a noisy quantum example oracle* $QEX^{\eta}(f, D)$ if the class is learnable in time polynomial in $1/(1-2\eta)$ as well as the standard parameters (we assume for simplicity that η is known). The probability $Pr_D[f = h]$ of success is taken over the random noise choices made by QEX^{η} as well as any random choices made by the learning algorithm.

We now state and prove the main result of this section.

Theorem 10 *DNF is quantum learnable with respect to uniform using a noisy quantum example oracle.*

Proof: Let $\mathbf{E}_{\eta}[f\chi_a]$ represent the expected value of $f\chi_a$ with respect to the uniform distribution on x given that $f(x)$ is produced by a noisy oracle (quantum or otherwise) having noise rate η. Using a technique due to Kearns [14], it can be shown that

$$\mathbf{E}_{\eta}[f\chi_a] = (1-2\eta)\mathbf{E}[f\chi_a].$$

Now consider running the quantum DNF algorithm developed earlier using a noisy quantum example oracle. The component of this algorithm impacted by the noisy oracle is the sampling of χ_a's to locate a weak approximator for f. Recall that when we used a noiseless oracle then we sampled each χ_a with probability proportional to $\mathbf{E}^2[f\chi_a]$. From the above expression we see that the effect of the noise is to reduce the expected values we see when using the noisy oracle by a value that is inverse polynomial in our allowed running time. Thus by increasing the number of samples by an appropriate amount we will still be able to find the desired χ_a's. □

Acknowledgments

The first author thanks Richard Cleve for an enlightening seminar on quantum computation.

References

[1] Aizenstein, H., Hellerstein, L., and Pitt, L. *Read-thrice DNF is Hard to Learn with Membership and Equivalence Queries.* in: **Proceedings of the 33rd Annual Symposium on Foundations of Computer Science.** 1992, pp. 523–532.

[2] Aizenstein, H. and Pitt, L. *Exact Learning of Read-k Disjoint DNF and Not-So-Disjoint DNF.* in: **Proceedings of the Fifth Annual Workshop on Computational Learning Theory**. 1992, pp. 71–76.

[3] Aizenstein, H. and Pitt, L. *Exact Learning of Read-Twice DNF Formulas.* in: **Proceedings of the 32nd Annual Symposium on Foundations of Computer Science**. 1991, pp. 170–179.

[4] Angluin, D., Frazier, M., and Pitt, L. *Learning Conjunctions of Horn Clauses.* in: **Proceedings of the 30th Annual Symposium on Foundations of Computer Science**. 1990, pp. 186–192.

[5] Bernstein, E. and Vazirani, U. *Quantum Complexity Theory.* in: **Proceedings of the 25th Annual ACM Symposium on Theory of Computing**. 1993, pp. 11–20.

[6] Blum, A., Furst, M., Jackson, J., Kearns, M., Mansour, Y., and Rudich, S. *Weakly Learning DNF and Characterizing Statistical Query Learning Using Fourier Analysis.* in: **Proceedings of the 26th Annual ACM Symposium on Theory of Computing**. 1994, pp. 253–262.

[7] Blum, A. and Rudich, S. *Fast Learning of k-Term DNF Formulas with Queries.* in: **Proceedings of the 24th Annual ACM Symposium on Theory of Computing**. 1992, pp. 382–389.

[8] Bruck, J. *Harmonic Analysis of Polynomial Threshold Functions.* **SIAM Journal of Discrete Mathematics**, vol. 3 (1990), pp. 168–177.

[9] Bshouty, N. H. *Exact Learning via the Monotone Theory.* in: **Proceedings of the 34th Annual Symposium on Foundations of Computer Science**. 1993, pp. 302–311.

[10] Freund, Y. *Boosting a Weak Learning Algorithm by Majority.* in: **Proceedings of the Third Annual Workshop on Computational Learning Theory**. 1990, pp. 202–216.

[11] Hancock, T. R. *Learning 2μDNF Formulas and $k\mu$ Decision Trees.* in: **Proceedings of the Fourth Annual Workshop on Computational Learning Theory**. 1991. pp. 199–209.

[12] Jackson, J. *An Efficient Membership-Query Algorithm for Learning DNF with Respect to the Uniform Distribution.* in: **Proceedings of the 35th Annual Symposium on Foundations of Computer Science**. 1994. *To appear.*

[13] Kearns, M., Li, M., Pitt, L., and Valiant, L. *On the Learnability of Boolean Formulae.* in: **Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing**. 1987, pp. 285–295.

[14] Kearns, M. J. *Efficient Noise-tolerant Learning from Statistical Queries.* in: **Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing. 1993, pp. 392–401.**

[15] Kushilevitz, E. and Mansour, Y. *Learning Decision Trees using the Fourier Spectrum.* **SIAM Journal on Computing**, vol. 22 (1993), pp. 1331–1348. *Earlier version appeared in* Proceedings of the Twenty Third Annual ACM Symposium on Theory of Computing, *pages 455–464, 1991.*

[16] Kushilevitz, E. and Roth, D. *On Learning Visual Concepts and DNF Formulae.* in: **Proceedings of the Sixth Annual Workshop on Computational Learning Theory**. 1993, pp. 317–326.

[17] Simon, D. R. *On the Power of Quantum Computation.* in: **Proceedings of the 35th Annual Symposium on Foundations of Computer Science**. 1994. *To appear.*

[18] Valiant, L. G. *A Theory of the Learnable.* **Communications of the ACM**, vol. 27 (1984), pp. 1134–1142.